PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

EX PARTE Shridhar Mukund et al.

Application for Patent

Filed December 2, 2003

Application No. 10/726,470

FOR:

NETWORKED PROCESSOR FOR A PIPELINED ARCHITECTURE

APPEAL BRIEF

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450 on April 22, 2009

Signed: ____________________
Jose M. Nunez

MARTINE PENILLA & GENCARELLA, LLP
Attorneys for Appellants

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is Adaptec, Inc., the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

The Appellants are not aware of any related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-20 are pending in the subject application. Claims 1-20 have been rejected and are on appeal.

IV. STATUS OF AMENDMENTS

Appellants submitted an amendment of September 15, 2006, in response to a non-Final Office Action mailed on May 15, 2006. This amendment was the last entered amendment. Subsequently, an Appeal Brief was filed on January 15, 2008, a non-Final Office Action was issued on June 27, 2008, a Request For Reconsideration was filed on October 1, 2008, and a Final Office Action was issued on January 7, 2009.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The subject invention is directed towards methods and apparatus for a processor configured to efficiently process incoming or outgoing packet data. The processor is arranged in a pipeline architecture, where one or more of the processors may be associated with a certain stage of the pipeline. The processor pipeline offloads previous processing performed by a central processing unit (CPU) of a host system, thereby freeing the CPU

for other processing to improve system performance. The processor is configured to allow a single cycle access to a large address space.

Accordingly, a networking application processor as recited in **claim 1** includes an input socket configured to receive data packets *(see Figures 4 and 5, and page 9, lines 8-20)* and a memory for storing instructions *(See Figures 4 and 5 and page 12 lines 16-21)*. The processor includes circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access of an operand from a memory location *(see Figures 6 and 7 and page 17, lines 15-25)*. An arithmetic logic unit (ALU) *(Figures 6 and 8 and page 19, lines 5-10)* and circuitry for aligning operands to be processed by the ALU, the circuitry for aligning operands causing the operands to be aligned by a lowest significant bit, wherein the circuitry for aligning the operands supplies an extension to the operands to allow the ALU to process different size operands *(see Figure 6 and page 23, lines 10-21)*.

Further, **claim 7** defines a processor that includes an input socket configured to receive data packets *(see Figures 4 and 5, and page 9, lines 8-20)* and a memory for storing instructions *(see Figures 4 and 5 and page 12 lines 16-21)*. The processor includes circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access from a memory location *(see Figures 6 and 7 and page 17, lines 15-25)*. An arithmetic logic unit (ALU) *(Figures 6 and 8 and page 19, lines 5-10)* is also included. The ALU is configured to receive a first and a second operand where the second operand is specified from an internal register *(see Figure 6 and page 18, lines 9-21)* and the first operand has a mask enabling the ALU to process a non-masked segment of the first operand *(see Figure 6 and page 21, lines 14-25)*.

Further yet, **claim 14** defines a processor capable of processing a data packet associated with a processing stage of a pipeline of processors. The processor includes a data random access memory (RAM) configured to enable access to data structures *(see Figures 5 and 6 and page 12, lines 14-20)*. Instruction fetch and decode circuitry configured to interpret instructions *(see Figures 5 and 6 and pages 13-14)* to be executed by an arithmetic logic unit (ALU) *(Figures 6 and 8 and page 19, lines 5-10)* is included. The instruction fetch and decode circuitry includes a read only memory (ROM) *(see Figures 5 and 6 and page 12, lines 14-20)*, the ROM configured to store code common to each processing stage associated with a pipeline of processors. The instruction and fetch decode circuitry includes a code RAM *(see Figures 5 and 6 and page 12, lines 14-20)* configured to download code specific to the processing stage and wherein the code specific to the processing stage is enabled for single cycle access. The processor includes instruction decode circuitry *(see Figures 5 and 6 and pages 13-14)* configured to recognize operating instructions. Execute and write back circuitry *(see Figures 5 and 6 and page 12, lines 14-20 and page 13, lines 3-15)* configured to set up operands to be processed by the ALU is included. The execute and write back circuitry includes internal registers *(see Figure 6 and page 18, lines 9-21)* for defining a first and a second operand. The processor further includes an arithmetic logic unit *(Figures 6 and 8 and page 19, lines 5-10)* for processing the first and second operands and align function circuitry *(see Figures 6 and 8 and pages 22-23)* for aligning the first and the second operands to be processed by the ALU. The align function circuitry causes the first and the second operands to be aligned by a lowest significant bit, wherein the align function circuitry supplies an extension to the each of the operands to allow the ALU to transparently process different size operands *(see page 23 lines 3-20)*.

As referred to in this application, each of the pipelined processors include both hardware such as, input socket interface, star processor, output socket interface, hardware accelerator, and relevant software to access the hardware. *(Figure 3 and related description on page 8, line 15 to page 11, line 5).* For the pipelined processors the output socket interface of a first processor is in communication with an input socket interface of a second processor, and so on *(see Figure 2).*

It should be appreciated that the above description represents only a summary of the present invention. A more in-depth discussion of the present invention is provided in the Detailed Description section of the application.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are presented for review:

A. Whether claims 1-6 and 14-20 are patentable under 35 USC 102(b) over Narayan et al. (U.S. Patent No. 5,822,559) (hereinafter "Narayan"); and

B. Whether claims 7-13 are patentable under 35 U.S.C. 103(a) over Narayan et al. (U.S. Patent No. 5,822,559).

VII. ARGUMENT

Appellants present the following arguments with respect to the rejected claims:

A. Rejection of claims 1-6 and 14-20 under 35 U.S.C. § 102(b) over Narayan et al.

1. <u>Claims 1 and 3-6</u>

i. Narayan does not disclose an input socket configured to receive data packets

Claim 1 defines an input socket configured to receive data packets. The Office has asserted that this feature is anticipated by Narayan because "the input data from I/O

module of the system intended to be used." While this statement may be true, this statement offers no information on how Narayan anticipates Appellants' claims, and the Office's rejection is improper.

Further, in the Response to Arguments section of the Office Action dated January 7, 2009, the Examiner has asserted the following:

> 'The limitation "an input socket configure to receive data packets" merely required a socket (any hardware for receiving data) which is able to receive data packets (any group of data). Inherently, any processor is going to send and receive data. Specifically, Narayan must perform I/O operations. These data received by the processor must be input into the processor with some sort of hardware. This hardware is "an input socket configure to receive data packets". Although the limitation is inherent in any data processor, Examiner gives a specific example of why it is in the system of Narayan for completeness and clarity" (page 10, second paragraph - emphasis added).

Appellant respectfully disagrees. First, not all processors are "going to send and receive data" [packets]. Many processors are not in a network environment and merely read and write operands from memory. Thus, not all processors inherently will have an input socket configured to receive data packets. Further, the Examiner asserts that a specific example was given, but Appellants were not able to find such specific example given in the rejection of claim 1. The Examiner has stated that the aforementioned feature is taught by Narayan because "the input data from I/O module of the system intended to be used" (page 2, last paragraph). The Examiner has not yet shown how Narayan teaches this feature. The Examiner has provided a generic statement as a rejection without identifying how Narayan teaches "the system intended to be used," nor the "I/O module." The Examiner has failed to identify how Narayan anticipates the Appellant's feature, and the Examiner has based the rejection exclusively on generic computer science principles, which Appellants believe are sometimes inaccurate. Thus, the Examiner has failed to

identify how Narayan teaches the aforementioned claim and has only offered conclusory statements with no analysis or considerations of the claim language.

ii. Narayan does not disclose circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access of an operand from a memory location

Claim 1 defines circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access of an operand from a memory location (emphasis added). The Office has not offered an explanation of how Narayan teaches the circuitry configured to access data structures, and has pointed to where Narayan supposedly teaches a memory location. Further yet, the Office has not put forward either an explanation on how Narayan anticipates a single cycle access of an operand in claim 1.

Further, in the Response to Arguments section of the Office Action dated January 7, 2009, the Examiner has asserted the following:

> 'The term "single cycle" is broad and merely requires the access be performed in a certain time period.' (page 11, last paragraph).

The assertion that '[t]he term "single cycle" is broad and merely requires the access be performed in a certain time period' is an untenable argument. The person skilled in the art would readily appreciate that the term "single cycle" refers to clock cycles, not to any time period. The Examiner has ignored basic computer science principles and made unreasonable generalizations, such as referring to a "single cycle" as "a certain time period." Further, Appellant's application clearly refers to clock cycles. See for example Figure 6 and the corresponding description, page 13, line 22, etc. Here again, the Examiner has only offered conclusory arguments having no rational underpinnings.

iii. Narayan does not disclose circuitry for aligning operands to be processed by the ALU, the circuitry for aligning operands causing the operands to be aligned by a lowest significant bit, wherein the circuitry for aligning the operands supplies an extension to the operands to allow the ALU to process different size operands

In addition, claim 1 defines circuitry for aligning operands to be processed by the ALU, the circuitry for aligning operands causing the operands to be aligned by a lowest significant bit, wherein the circuitry for aligning the operands supplies an extension to the operands to allow the ALU to process different size operands (emphasis added). Instead of teaching how Narayan teaches this limitation, the Office has explained how Narayan teaches a different limitation, by changing the language of the claim and describing instead how Narayan teaches "causing the operand" (in singular) "to be aligned" (see page 4, line 1 of Office Action dated 6/27/2008) and "circuitry for aligning the operand" (in singular) (see page 4, line 11 of Office Action dated 6/27/2008). Under a 102 rejection, the Office must state how all of the claim limitations are anticipated by the prior art. Changing the Applicants' claim language is inappropriate and misleading. Even assuming *arguendo* that Narayan taught circuitry of aligning the operand, this does not anticipate circuitry for aligning the operands, as claimed by Applicants. Applicants respectfully request that the Office clearly explains how the potential prior art teaches Applicants claims using the exact language of the claims, or using obviousness arguments to explain the differences.

Further, in the Response to Arguments section of the Office Action dated January 7, 2009, the Examiner has asserted the following:

"Examiner disagrees. The lack of an "s" on operands is clearly a typographic error. The two operands (1985 and 56 in the example shown) are clearly aligned to each other therefore two operands

> are aligned. Additionally, any circuitry for aligning (as claimed) one operand can clearly align more than one. It would be foolish to assume that only one instruction would flow thorough a processor. Additionally, the claim language does not require the operands to be aligned, it merely requires circuitry that is able to align operands" (page 12, third paragraph); and
>
> "Examiner disagrees. As stated above, not only does the prior art teach aligning 2 operands, the claim merely requires circuitry capable of aligning multiple operands (which is inherent of any circuitry that can align 1 operand)" (page 12, last paragraph).

Appellants respectfully disagree. Appellants believed that the Examiner's assertion was not a typographical error because Narayan teaches aligning one operand, that is, that "the B <u>operand</u> can be aligned ... or scaled ... or complemented" (col. 115, lines 8-10 - emphasis added). Additionally, the Examiner is referring to an example created by the Examiner, which is irrelevant, since the Examiner must show how the prior art teaches Appellants' features, and not writing up examples that supposedly teach Appellants' features. Nevertheless, taking into consideration that this was a typographical error, Appellant asserts that Narayan does not teach the aforementioned limitation because Narayan only teaches aligning <u>one</u> operand instead of circuitry for aligning operands. Further, the Examiner has asserted that "any circuitry for aligning ... one operand can clearly align more than one," which may be true but offers an unreasonable interpretation of the claimed language. Appellants believe that the Examiner refers to aligning one operand in one cycle, then aligning one operand later in time, etc. However, the person skilled in the art would readily appreciate that aligning operands refers to aligning operands at the same moment in time, so both operands can be operated on simultaneously, as shown by the claim language "causing the <u>operands</u> to be aligned by a lowest significant bit."

Further still, the Examiner asserts that "the claim merely requires circuitry capable of aligning multiple operands (which is <u>inherent of any circuitry that can align 1 operand</u>)."

As previously discussed, the Examiner is not considering the claims as a whole and provides an unreasonable interpretation of the term "circuitry for aligning operands" by implying that more than one operand can be aligned by one circuit if you wait for the circuit to perform multiple operations. The Examiner has ignored that the circuitry for aligning operands causes the operands to be aligned by a lowest significant bit, meaning that both operands have to be aligned at the same time.

For all these reasons, Appellant believes that the prior art does not teach circuitry for aligning operands to be processed by the ALU, the circuitry for aligning operands causing the operands to be aligned by a lowest significant bit.

2. <u>Claims 14-20</u>

i. Narayan does not disclose that the code RAM configured to download code specific to the processing stage and wherein the code specific to the processing stage is enabled for <u>single cycle access</u>

Claim 14 defines that the code RAM configured to download code specific to the processing stage and wherein the code specific to the processing stage is enabled for single cycle access. This feature is believed to be patentable for at least the same reasons described hereinabove for claim 1 with respect to the claimed feature including circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling <u>a single cycle access</u> of an operand from a memory location.

ii. Narayan does not disclose align function circuitry for aligning the first and the second operands to be processed by the ALU, the align function circuitry the circuitry causing the first and the second operands to be aligned by a lowest significant bit

Claim 14 defines align function circuitry for aligning the first and the second operands to be processed by the ALU, the align function circuitry the circuitry causing the first and the second operands to be aligned by a lowest significant bit. This feature is

believed to be patentable for at least the same reasons described hereinabove for claim 1 with respect to the claimed feature including circuitry for aligning operands causing the operands to be aligned by a lowest significant bit.

iii. Narayan does not disclose a processor capable of processing a data packet associated with a processing stage of a pipeline of processors

Claim 14 defines a processor capable of processing a data packet associated with a processing stage of a pipeline of processors. The Office has asserted that Narayan teaches the pipeline of processors because "each pipeline stage does processing and therefore any pipelined processor contains a 'pipeline of processors,'" and pointed to the following excerpt:

> "As used herein, a 'clock cycle' is an interval of time accorded to various stages of an instruction processing pipeline within the microprocessor to complete their various functions. Storage devices (e.g. registers and arrays) capture their values according to a clock signal defining the clock cycle." (col. 1, lines 15-20 - emphasis added).

Appellants respectfully disagree. Narayan teaches an instruction processing pipeline within the microprocessor, which nowhere suggests a pipeline of processors. Suggesting that any pipelined processor contains a pipeline of processors is simply illogical and untrue. The Office is offering a twisted play on words, which any reasonable person skilled in the art would consider inappropriate. Not everything in a system that performs a function is a processor. For example, memory performs a function of storing data, yet a person skilled in the art would never call a memory a processor. The Office has failed to show how Narayan teaches a processor capable of processing a data packet associated with a processing stage of a pipeline of processors, and the Office's rejection is improper.

Further, in the Response to Arguments section of the Office Action dated January 7, 2009, the Examiner has asserted the following:

> 'Examiner disagrees. The definition of a processor is "A part of a computer, such as the central processing unit, that performs calculations or other manipulations of data" ... A processing stage is by definition, a processor, not because of a "twisted play on words" as applicant alleges' (page 13, third paragraph - emphasis added).

Appellants respectfully disagree. Using the same reference as used by the Examiner, Dictionary.com defines stage as "*a single step or degree in a process; a particular phase, period, position, etc., in a process, development, or series.*" If a processor is "a part of a computer" (as cited by the Examiner) then, which part of the computer would the stage be? "A part of a computer" denotes that a processor is something tangible. However, a stage is not tangible because it refers to a "step or degree in a process," and a process is something intangible. It is not possible to touch or hold a "step or degree in a process." Thus, if a stage is not tangible, a stage cannot be a part of a computer and cannot be a processor.

iv. Narayan does not disclose that the instruction fetch and decode circuitry includes a read only memory (ROM), the ROM configured to store code common to each processing stage associated with a pipeline of processors

Further, claim 14 defines that the instruction fetch and decode circuitry includes a read only memory (ROM), the ROM configured to store code common to each processing stage associated with a pipeline of processors. The Office has asserted that the ROM is anticipated by Narayan in "portion of the MROM" (emphasis added). Appellants respectfully disagree. The Office has once more used vague language such as "portion" that fails to explain how the prior art teaches Appellants' claims. Moreover, Narayan teaches that "MROM unit 209 parses and serializes the instruction into a subset of defined fast path instructions to effectuate a desired operation" (col. 6, lines 29-31). Thus, a MROM does not suggest a ROM, although the names sound similar, because the MROM unit parses and serializes instructions, and a Read Only Memory stores bits and does not

parse and serialize instructions. Thus, Narayan does not teach that the instruction fetch and decode circuitry includes a read only memory (ROM), the ROM configured to store code common to each processing stage associated with a pipeline of processors.

Further, in the Response to Arguments section of the Office Action dated January 7, 2009, the Examiner has asserted that "MROM units decode instructions and using the table output micro-operations using a lookup table. That table is a ROM." Appellants respectfully disagree. The Examiner's response is vague because it refers to "**a** look up table" and "**that** table" (emphasis added). The Examiner does not identify which lookup table the MROM is using, or where is the text describing the table in the prior art, or why is that table a Read Only Memory. The table is not described in the excerpts cited by the Examiner and the Appellant has not been able to find it with such vague description. The rejection is conclusory and lacks any rational underpinning.

3. Claim 2

i. Narayan does not disclose the feature wherein the single cycle access enables the data to be addressed and operated on in a single clock cycle without being placed into a register

Dependent claim 2 defines the feature wherein the single cycle access enables the data to be addressed and operated on in <u>a single clock cycle</u> without being placed into a register (emphasis added). The Office has asserted that this feature is taught by Narayan in col. 10, last paragraph (excerpted below). Appellants respectfully disagree.

> "During certain clock cycles, instructions may not be fetched by fetch control unit 266. Instead, the valid instructions within <u>instruction bytes fetched</u> during a previous clock cycle may not have been completely transferred to storage devices 254, as indicated by instruction group control unit 260. As used herein, the term "valid instruction" refers to an instruction within the fetched instruction bytes which is intended to be dispatched to decode units 208" (col. 10, last paragraph - emphasis added).

Additionally, the Examiner has asserted the following:

> "Note that if an <u>instruction is fetched</u>, the <u>data contained within that instruction</u> is addressed (with a program counter) and <u>operated on</u> (read from memory). The definition of the word "operate" according to The American Heritage® Dictionary of the English Language, Fourth Edition is to 'perform a function: work'. Under this definition, a read (fetch) from memory is reasonably considered to be an operation" (page 6, second paragraph - emphasis added).

The Examiner has failed to consider the claim as a whole, and it seems like the Examiner is trying to redefine the very basics of computer science, as understood by the person skilled in the art. First, the excerpt cited by the Examiner nowhere suggests that the data to be addressed and operated on in a <u>single clock cycle</u> (emphasis added). Appellants respectfully request that if the Examiner maintains this rejection, the Examiner explains where Narayan teaches that the operation is performed in a single clock cycle, so Appellants can clearly explain in the Appeal Brief how Narayan does not teach this feature.

Additionally, claim 2 is referring to accessing data structures because claim 1 defines access[ing] <u>data structures</u> (emphasis added). The Examiner equates an instruction with data because the "data [is] contained within that instruction." The Examiner is playing with the meaning of the word "data" and ignoring that Appellants are claiming "data structures." Thus, fetching instructions does not teach accessing data structures because an instruction is not the same as a data structure, and the Examiner's rejection is improper. Still yet, the Examiner's assertion that reading an instruction from memory is the same as executing an instruction is conclusory and completely in conflict with the basic principles of operation of a processor system. For all these reasons, Appellants assert that Narayan does not anticipate that the single cycle access enables the data to be addressed and operated on in a single clock cycle without being placed into a register, as claimed by Appellants.

Additionally, in the Response to Arguments section of the Office Action dated January 7, 2009, the Examiner has asserted that "[s]imply pulling the instruction (fetching) is operating on it" (page 15, last paragraph). Appellant respectfully disagrees, for the reasons described hereinabove, and further asserts that a person skilled in the art would not reasonably interpret pulling an instruction is operating on it.

B. Rejection of claims 7-13 under 35 U.S.C. 103(a) over Narayan et al.

1. <u>Claims 7 and 9-13</u>

i. Adding obviousness reasoning to the 102 rejection based on Narayan does not cure the deficiencies of the Narayan reference relative to claim 1

Claim 7 is believed to be patentable for at least the same reasons with respect to claim 1 described hereinabove. More specifically, the following features in claim 7 are believed to be patentable for the same reasons described previously with reference to claim 1:

- an input socket configured to receive data packets;

- circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access from a memory location; and

- align function circuitry for aligning the first and the second operands to be processed by the ALU, the align function circuitry the circuitry causing the first and the second operands to be aligned by a lowest significant bit.

Additionally, dependent claims 9-13 are submitted to be patentable for at least the same reasons independent claim 7 is believed to be patentable.

2. <u>Claim 8</u>

i. Narayan does not disclose the feature wherein the single cycle access enables the data to be addressed and operated on in a single clock cycle without being placed into a register

Claim 8 further defines the single cycle access and is patentable over Narayan for at least the above stated reasons with reference to claim 2.

C. Conclusion

In view of the foregoing reasons, the Appellants submit that each of the claims 1-20 are patentable. Therefore, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's rejections of the claims on appeal.

Respectfully submitted,
MARTINE PENILLA & GENCARELLA, LLP

Jose M. Nunez
Reg. No. 59,979

710 Lakeway Drive, Suite 200
Sunnyvale, CA 94085
Telephone: (408) 749-6900
Facsimile: (408) 749-6901
Customer Number 25920

VIII. CLAIMS APPENDIX

1. A networking application processor, comprising:

an input socket configured to receive data packets;

a memory for storing instructions;

circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access of an operand from a memory location;

an arithmetic logic unit (ALU); and

circuitry for aligning operands to be processed by the ALU, the circuitry for aligning operands causing the operands to be aligned by a lowest significant bit, wherein the circuitry for aligning the operands supplies an extension to the operands to allow the ALU to process different size operands.

2. The networking application processor of claim 1, wherein the instructions have a width of 96 bits, and wherein the single cycle access enables the data to be addressed and operated on in a single clock cycle without being placed into a register.

3. The networking application processor of claim 1, wherein the different size operands are selected from the group consisting of 8 bit operands, 16 bit operands, and 32 bit operands.

4. The networking application processor of claim 1, further including:

an output socket for transmitting processed data; and

a 64 bit bus connecting the input socket and the output socket.

5. The networking application processor of claim 1, wherein the extension to the operand fills each higher bit with a value.

6. The networking application processor of claim 1, wherein the operand is selected from the group consisting of a source operand, a destination operand, an immediate operand, and an internal register operand.

7. A processor, comprising:

an input socket configured to receive data packets;

a memory for storing instructions;

circuitry configured to access data structures associated with a processing stage, the circuitry configured to access data structures enabling a single cycle access from a memory location; and

an arithmetic logic unit (ALU), the ALU configured to receive a first and a second operand; the second operand being specified from an internal register, the first operand having a mask enabling the ALU to process a non-masked segment of the first operand.

8. The processor of claim 7, wherein the instructions have a width of 96 bits, and wherein the single cycle access enables the data to be addressed and operated on in a single clock cycle without being placed into a register.

9. The processor of claim 7, wherein each of the instructions include a loadback feature enables random accesses to one of a source indirect register or a destination indirect register through indirect addressing.

10. The processor of claim 7, wherein the mask is associated with an immediate value of the first operand.

11. The processor of claim 7, wherein the first and the second operands are associated with a size selected from the group consisting of 8 bit operands, 16 bit operands, and 32 bit operands.

12. The processor of claim 7, wherein the first operand is selected from the group consisting of a source operand, a destination operand, an immediate operand, and an internal register operand.

13. The method of claim 7, wherein the memory location is a static random access memory (SRAM).

14. A processor capable of processing a data packet associated with a processing stage of a pipeline of processors, the processor comprising:

a data random access memory (RAM) configured to enable access to data structures;

instruction fetch and decode circuitry configured to interpret instructions to be executed by an arithmetic logic unit (ALU) , the instruction fetch and decode circuitry including,

a read only memory (ROM), the ROM configured to store code common to each processing stage associated with a pipeline of processors;

a code RAM, the code RAM configured to download code specific to the processing stage and wherein the code specific to the processing stage is enabled for single cycle access; and

instruction decode circuitry configured to recognize operating instructions;

execute and write back circuitry configured to set up operands to be processed by the ALU, the execute and write back circuitry including,

internal registers for defining a first and a second operand;

an arithmetic logic unit for processing the first and second operands; and

align function circuitry for aligning the first and the second operands to be processed by the ALU, the align function circuitry the circuitry causing the first and the second operands to be aligned by a lowest significant bit, wherein the align function circuitry supplies an extension to the each of the operands to allow the ALU to transparently process different size operands.

15. The processor of claim 14, wherein the extension to each of the operands fills each higher bit with a value.

16. The processor of claim 14, wherein the different size operands have a width selected from the group consisting of 8 bits, 16 bits, and 32 bits.

17. The processor of claim 14, wherein the operating instructions wherein the operating instructions are formatted as 96 bit instructions, each of the 96 bit instructions including a single return bit.

18. The processor of claim 14, wherein the processor is configured as a two stage pipeline for pipelining an instruction fetch and decode operation and an execute and write back operation.

19. The processor of claim 14, wherein the operating instructions include microcode configured to predict a likely direction for a branch instruction.

20. The processor of claim 19, wherein no operation (NOP's) instructions are included, the NOP's configured block an invalidated pre-fetched instruction.

IX. EVIDENCE APPENDIX

There is currently no evidence entered and relied upon in this Appeal.

X. RELATED PROCEEDINGS APPENDIX

There are currently no decisions rendered by a court or the Board in any proceeding identified in the Related Appeals and Interferences section.